

Estructuras

Programación I

Lic. Mauro Gullino maurogullino@gmail.com

UTN FRH

Struct

Es un conjunto de variables del mismo o distinto tipo, que se trabajan en grupo, e individualizada cada una por nombres ("campos").

```
struct empleado
```

Nombre: _____

Cargo: _____

DNI: _____

Sintaxis

```
struct alumno {    //declaración de tipo
    int legajo;
    char sexo;
};
```

```
struct alumno pepe; //declaración de var
pepe.legajo = 123; //asignación
pepe.sexo = 'M';
```

Sintaxis (2)

```
struct {    //declaración de tipo y var
    int legajo;
    char sexo;
} pepe;
```

```
pepe.legajo = 123; //asignación
pepe.sexo = 'M';
```

Tipos del usuario

Son tipos de datos definidos por el usuario, y pasan a formar parte del conjunto de tipos de nuestro programa

Declararemos variables con nuestros "nuevos tipos"

Se pueden copiar las variables (!= vectores)

Copia de variables struct

```
struct alumno {  
    int legajo;  
    char sexo;  
};
```

```
struct alumno pepe, juan;  
pepe.legajo = 123;  
pepe.sexo = 'M';  
juan = pepe;  // !!
```

Tamaño de la estructura

```
/* no se puede considerar el tamaño de la  
struct como la suma de sus campos */
```

```
struct {  
    int A;  
    char B;  
    int C;  
} prueba;
```

```
printf("%d", sizeof(prueba));
```

Copia de vectores con struct

```
struct vector {  
    int nums[10];  
};
```

```
struct vector vec1, vec2;  
int i;  
for (i=0; i<10; i++) vec1.nums[i]=i;  
  
vec2 = vec1;    // legal
```


Pasaje a funciones

Al ser variables se pueden pasar a y retornar de funciones "por valor".

Debe considerarse:

- lugar a ocupar en la pila de la función
- tipo de transferencia

Se puede hacer un pasaje "por referencia" con variables tipo "puntero" (*Programación 2*)

Ejemplo de pasaje y retorno

```
#include <stdio.h>

struct prueba { int i; float j; };

int main() {
    struct prueba p, q;
    p.i = 7;    p.j = 10.40;
    q = dupli(p);
}

struct prueba dupli(struct prueba t) {
    t.i = t.i * 2; t.j = t.j * 2;
    return t;
}
```

Ejemplo de fecha

```
#include <stdio.h>

struct fecha { int dia; int mes; int anio; };

int main() {
    struct fecha hoy;
    printf("Ingresa el dia de hoy. dd/mm/yyyy: ");
    scanf("%d/%d/%d", &hoy.dia, &hoy.mes, &hoy.anio);
}

//notar el ampersand &
```

Ejercicio

Realice una función que reciba dos struct fecha y retorne cuál es la fecha mayor.

```
int fecha_mayor(struct fecha, struct fecha);  
// 1 si la primera es mayor  
// -1 si la segunda es mayor  
// 0 si son iguales
```

Estructuras anidadas

```
struct fecha { int dia; int mes; int anio; };  
struct alumno {  
    char nombre[40];  
    struct fecha fnacim;  
    float promedio;  
};
```

```
struct alumno pepe;  
strcpy(pepe.nombre, "Pepe Gomez");  
pepe.fnacim.dia=1; pepe.fnacim.mes=2;  
pepe.fnacim.anio=1980;  
pepe.promedio=7.8;
```

Ejercicio

Realice una función que reciba dos struct alumno y retorne cuál es el de mayor edad (el más viejo). Utilizar la función `fecha_mayor` del otro ejercicio.

```
int alumno_mayor(struct alumno, struct alumno);  
// 1 si el primero es mayor  
// -1 si el segundo es mayor  
// 0 si son iguales
```

Vectores de estructuras

```
struct alumno {  
    char nombre[40]; float promedio;  
};
```

```
struct alumno alumnos[5];
```

```
// alumnos[0].promedio  
// alumnos[1].nombre  
// alumnos[2].nombre[2]
```



The diagram shows a stack of five identical rectangular forms, slightly offset to the right and bottom from each other, creating a 3D effect. The topmost form contains the text 'Nombre: _____' and 'Promedio: _____' with horizontal lines for input.

Nombre: _____
Promedio: _____

Pasaje a función

```
struct alumno {  
    char nombre[40]; float promedio;  
};
```

```
float prom_general(struct alumno todos[], int n) {  
    float prom, acum=0; int i;  
    for (i=0; i<n; i++) {  
        acum += todos[i].promedio;  
    }  
    prom = acum / n;  
    return prom;  
}
```


Ejercicio

Dada una estructura "alumno", con los campos "nombre" y "promedio":

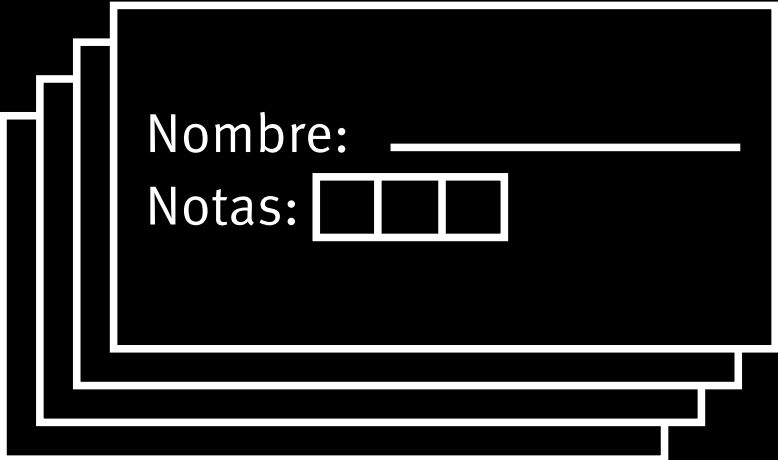
- a) construya una función main que permita el ingreso de un vector de alumnos.
- b) construya una función que recibe un vector de alumnos y los muestra en pantalla.
- c) construya una función que reciba un vector de alumnos y los ordene descendente por promedio.

¿Cómo sería así?

```
struct alumno {  
    char nombre[40]; float notas[3];  
};
```

```
struct alumno alumnos[5];
```

```
// alumnos[0].promedio  
// alumnos[1].nombre  
// alumnos[2].notas[0]
```



The diagram shows a stack of five overlapping rectangular boxes, representing a memory array of student records. The top-most box contains the following text and input fields:

Nombre: _____

Notas:

Ejercicio

Dada una estructura "alumno", con los campos "nombre" y "notas" (vector de notas):

- a) construya una función main que permita el ingreso de un vector de alumnos.
- b) construya una función que reciba un vector de alumnos y un "código de materia" y muestre los alumnos ordenados por la nota obtenida en esa materia
- c) construya una función que reciba un vector de alumnos y los ordene descendente por promedio.